1. **SQL:**
   a. Way to interact with a database
   b. Structured Query language
   c. commands typically terminated by ;

2. **Basic operations**
   a. Database-level
      i. `CREATE DATABASE db_name;`
      ii. `SHOW DATABASES;`
      iii. `DROP DATABASE db_name;`
      iv. `SHOW WARNINGS;`

   b. Table level
      i. `DESCRIBE table;`
      ii. `CREATE TABLE table_name (field1_name TYPE(SIZE), filed2_name TYPE(SIZE),...);`
      iii. `DROP TABLE table_name;`
      iv. `SHOW TABLES;`
      v. `ALTER TABLE;`
      vi. `DESCRIBE table;`

   c. Data level
      i. `LOAD DATA LOCAL INFILE "inile.txt" INTO TABLE table_name;`
      ii. `INSERT INTO table_name VALUES ('MyName', 'MyOwner', '2002-08-31');`
      iii. `SELECT column_name FROM table;`
      iv. `SELECT DISTINCT column_name FROM table;`
      v. `SELECT col1, col2 FROM table ORDER BY col2;`
      vi. `SELECT CURRENT_DATE, (YEAR(CURRENT_DATE)-YEAR(date_col)) AS time_diff [FROM table];`
      vii. `SELECT * FROM table WHERE rec LIKE "blah%";`
      viii. `SELECT * FROM table WHERE rec RLIKE "^b$";`
           (. for char, [... for char class, * for 0 or more instances ^ for beginning, (n) for repeat n times, and $ for end) (RLIKE or REGEXP) To force case-sensitivity, use "REGEXP BINARY"
      ix. `SELECT COUNT(*) FROM table;`
      x. `SELECT owner, COUNT(*) FROM table GROUP BY owner;`
         (GROUP BY groups together all records for each 'owner')
      xi. `DELETE FROM table <deletes data in table—keeps the table>
      xii. `UPDATE table SET column_name = "new_value" WHERE record_name = "value";`
      xiii. `ALTER TABLE tbl ADD COLUMN [column_create syntax] AFTER col_name;`
      xiv. `ALTER TABLE tbl DROP COLUMN col;`
d. Data Types
   i. Numeric (bold and underlined are in SQL spec)
      1. exact: TINYINT, SMALLINT, INT, INTEGER, BIGINT, BIT, DECIMAL, NUMERIC, DEC.
      2. approximate: REAL, DOUBLE, DOUBLE PRECISION, FLOAT
   ii. Date formats: DATE, TIME, TIMESTAMP, DATETIME, YEAR
   iii. Character String Types: CHAR, VARCHAR, TEXT, TINYTEXT, MEDIUMTEXT, LONGTEXT, ENUM, SET
   iv. Binary String/data types: BLOB, TINYBLOB, MEDIUMBLOB, LONGBLOB, BINARY, VARBINARY

e. Data Type options & notes:
   i. Integer Options: (length) [UNSIGNED] [ZEROFILL]
      1. length determines how much SPACE is used to display the integer it has
         no affect on minimum or maximum values.
      2. If ZEROFILL is used the ‘spaces’ will be replaced with zeros. (NOTE: using integers longer than the display width MAY cause problems in certain operations).
      EX: TINYINT(4)

<table>
<thead>
<tr>
<th>Type</th>
<th>Bytes</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TinyInt</td>
<td>1</td>
<td>-128</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>255</td>
</tr>
<tr>
<td>SmallInt</td>
<td>2</td>
<td>-32768</td>
<td>32767</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>65535</td>
</tr>
<tr>
<td>MediumInt</td>
<td>3</td>
<td>-8388608</td>
<td>8388607</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>16777215</td>
</tr>
<tr>
<td>Int</td>
<td>4</td>
<td>-2147483648</td>
<td>2147483647</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>4294967295</td>
</tr>
<tr>
<td>BigInt</td>
<td>8</td>
<td>-9223372036854775808</td>
<td>9223372036854775807</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>18446744073709551615</td>
</tr>
</tbody>
</table>

ii. Bit wise sequences: BIT (same options as the integers), but length really does mean something (it can range from 1-64 in MySQL 6.0)
   1. b'value' notation: b'111' and b'10000000' represent 7 and 128, respectively.
   2. If you assign a value to a BIT(M) column that is less than M bits long, the value is padded on the left with zeros. For example, assigning a value of b'101' to a BIT(6) column is, in effect, the same as assigning b'000101'.

iii. Floating point:
   1. MySQL uses four bytes for single-precision values and eight bytes for double-precision values.
2. Under SQL spec FLOAT(22) will use 22 bits for the sig-figs
3. mySQL ignores this: 0-23 uses 4 bytes and 24-53 is an eight-byte double-precision.
4. mySQL has non-standard (length,decimal) option: FLOAT(m,d) means:
   a. stored with up to M digits, d is number after decimal—SQL will round upon insertion
   b. For example, a column defined as FLOAT(7,4) will look like – 999.9999 when displayed. MySQL performs rounding when storing values, so if you insert 999.00009 into a FLOAT(7,4) column, the approximate result is 999.0001.
   c. UNSIGNED: Prevents negative value, but doesn’t change highest possible value (contrast with integers)
   d. Example using DECIMAL:
       salary DECIMAL(5,2)

   In standard SQL, the syntax DECIMAL(M) is equivalent to DECIMAL(M,0). (Default value of M is 10)

5. Character String [CHARACTER SET charset_name] [COLLATE collation_name]
   a. These strings have character sets assigned to them (this affects how sorting is done)
   b. And can also be given varying collations
6. Binary strings: BINARY(length), VARBINARY(length)
   Binary strings have no character set, and sorting and comparison are based on the numeric values of the bytes in the values.
7. Binary Large Objects & TEXT: BLOB, TINYBLOB, MEDIUMBLOB, LONGBLOB, TEXT [BINARY], LONGTEXT [BINARY], TINYTEXT[BINARY], MEDIUMTEXT[BINARY]
   a. Blobs can hold variable amount of binary data
   b. Text can hold variable amount of character data
   c. Storage requirements vary for engine-types
8. ENUM(value1,value2,value3,...)
   a. List of allowed character strings:
      CREATE TABLE sizes ( 
         name ENUM('small', 'medium', 'large') 
      );
   b. Values can be accessed using the character strings, or the corresponding indices (index of a null value is NULL)

<table>
<thead>
<tr>
<th>Value</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>' '</td>
<td>0</td>
</tr>
<tr>
<td>'one'</td>
<td>1</td>
</tr>
<tr>
<td>'two'</td>
<td>2</td>
</tr>
<tr>
<td>'three'</td>
<td>3</td>
</tr>
</tbody>
</table>

An enumeration can have a maximum of 65,535 elements.

9. SET(value1,value2,value3,...)
   [CHARACTER SET charset_name] [COLLATE collation_name]
   Can have 0 or more values which can be chosen from the list of acceptable values.

Some examples with sets:

```sql
mysql> CREATE TABLE myset (col SET('a', 'b', 'c', 'd'));
```

If you insert the values 'a,d', 'd,a', 'a,d,d', 'a,d,a', and 'd,a,d':

```sql
mysql> INSERT INTO myset (col) VALUES
      -> ('a,d'), ('d,a'), ('a,d,a'), ('a,d,d'), ('d,a,d');
```

Then all of these values appear as 'a,d' when retrieved:

```sql
mysql> SELECT col FROM myset;
```

```
<table>
<thead>
<tr>
<th>col</th>
</tr>
</thead>
<tbody>
<tr>
<td>a,d</td>
</tr>
<tr>
<td>a,d</td>
</tr>
<tr>
<td>a,d</td>
</tr>
<tr>
<td>a,d</td>
</tr>
<tr>
<td>a,d</td>
</tr>
</tbody>
</table>
```

5 rows in set (0.04 sec)

```sql
mysql> SHOW WARNINGS;
```

```
+---------+------+------------------------------------------+
| Level   | Code | Message                                  |
|---------+------|------------------------------------------|
| Warning | 1265 | Data truncated for column 'col' at row 1 |
+---------+------|------------------------------------------+
1 row in set (0.04 sec)
b.  spatial_type:  We’ll ignore this

2.  Understanding QUERIES

drop table if exists table_A;
create table table_A (id INT NOT NULL primary key AUTO_INCREMENT, x FLOAT, y FLOAT);
insert into table_A (x,y) VALUES (rand(),rand());
select * from table_A;
drop table if exists table_B;
create table table_B (id INT NOT NULL primary key AUTO_INCREMENT, x FLOAT, y FLOAT);
insert into table_B (x,y) VALUES (rand(),rand());
select * from table_B;

**Conditional selection:**
select * from table_A where x > .2;

**individual columns:**
select id,x from table_A;

**column aliases:**
select id, x as new_x from table_A;

**pandemonium joins:**
select * from table_A,table_B;
select * from table_A as a,table_B as b;
select a.* from table_A as a,table_B as b;
select * from table_A,table_B,table_A;

**Specifying columns from multiple tables**
select id,a.x,a.y from table_A as a,table_B as b;
select * from table_A,table_B,table_A as c;

**Smarter Join:**
select * from table_A as a,table_B as b where a.id=b.id;
select * from table_A as a,table_B as b where a.id>b.id;
select * from table_A as a,table_B as b where a.id>=b.id;
select a.id as ident, a.x as old_x, b.x as new_x from table_A as a, table_B as b where ident=b.id;
select a.id as ident, a.x as old_x, b.x as new_x from table_A as a, table_B as b where a.id=b.id;
drop table if exists table_C;
create table table_C (id INT NOT NULL primary key AUTO_INCREMENT, x FLOAT, y FLOAT, timestamp TIMESTAMP);
insert into table_C (x,y) values(rand(),rand());
select * from table_C where id < 5 and id >= 2;
delete from table_C where id < 5 and id >= 2;
insert into table_C (x,y) values(rand(),rand());
select * from table_C;
select * from table_C order by timestamp;
select * from table_A as a,table_B as b,table_C as c where a.id=b.id and b.id=c.id;
insert into table_A (x,y) values(rand(),rand());
insert into table_B (x,y) values(rand(),rand());
insert into table_B (x,y) values(rand(),rand());
insert into table_B (x,y) values(rand(),rand());
select * from table_A as a,table_B as b,table_C as c where a.id=b.id and b.id=c.id and a.id=5;
deleting from multiple tables at once:
delete a,b from table_A as a,table_B as b,table_C as c where a.id=b.id and b.id=c.id and a.id=5;
select * from table_C;
select * from table_B;
select * from table_A;
select * from table_A as a,table_B as b,table_C as c where a.id=b.id and b.id=c.id;
insert into table_A (x,y) values(rand(),rand());
insert into table_B (x,y) values(rand(),rand());
insert into table_B (x,y) values(rand(),rand());
insert into table_B (x,y) values(rand(),rand());
select * from table_A as a,table_B as b,table_C as c where a.id=b.id and b.id=c.id;
select * from table_C where id > 5;
update table_C set x=2 where id > 5;
select * from table_C;
# You can use the 'ORDER BY' clause on update to control the order in which the columns are updated (this can be important)
select * from table_A as a,table_B as b, table_C as c where a.id=b.id and b.id=c.id;
update table_A as a,table_B as b, table_C as c set a.y=3,b.x=0 where a.id=b.id and b.id=c.id;
select * from table_A as a, table_B as b, table_C as c where a.id=b.id and b.id=c.id;
Some string manipulation:
create table string_land (word_a char(20), word_b char(20));
load data infile 'random.txt' into table string_land;
select * from string_land;
delete from string_land;
load data local infile 'random.txt' into table string_land fields terminated by ' ';
select * from string_land;
load data local infile 'random.txt' into table string_land fields terminated by ' ' lines terminated by "; 
select * from string_land;
select upper(word_A) from string_land limit 10;
select substring(word_A,1,1) from string_land limit 10;
select substring(word_A,2,2) from string_land limit 10;
Notice that it’s (start, length)